



CONSEJERÍA DE EDUCACIÓN
Comunidad de Madrid

IES ENRIQUE TIERNO GALVÁN
Parla

**CFGS DESARROLLO DE APLICACIONES
MULTIPLATAFORMA**
Curso 2024/2025

Proyecto DAM

TÍTULO: Aplicación gestión de videojuegos.

Alumno: David Bargalló Ortiz.

Tutor: Julián Parra Perales.

Junio de 2025

Índice

Contexto de la aplicación	2
Mundo real del problema	2
Qué aplicaciones existen.....	2
Mejoras de la aplicación respecto a las existentes	2
Casos de Uso - F	2
Requisitos funcionales y no funcionales	6
Funcionales.....	6
No funcionales.....	6
Diseño.....	7
GUI.....	7
UI (vistas).....	7
UX (usabilidad)	10
Diagrama navegación	11
Reutilización (fragmentos futuros)	11
Arquitectura	11
Despliegue.....	11
Componentes	12
Base de datos	13
Paquetes, Interfaces y Clases	14
Plan de pruebas (cómo)	14
Implementación	14
Entorno de Desarrollo	14
Implantación/Puesta en producción.....	15
Capturas de la ejecución de la funcionalidad.....	15
Información sobre la versión y software necesario	20
Elementos destacables del desarrollo.....	20
Conclusiones	21
Bibliografía	21
Anexos	21

Contexto de la aplicación

Mundo real del problema

Muchos jugadores acumulan videojuegos en diferentes formatos y plataformas, lo que dificulta su organización. Esta aplicación busca centralizar toda esta información en un único lugar, resolviendo problemas como la desorganización, dificultad de localización, y falta de estadísticas.

Qué aplicaciones existen

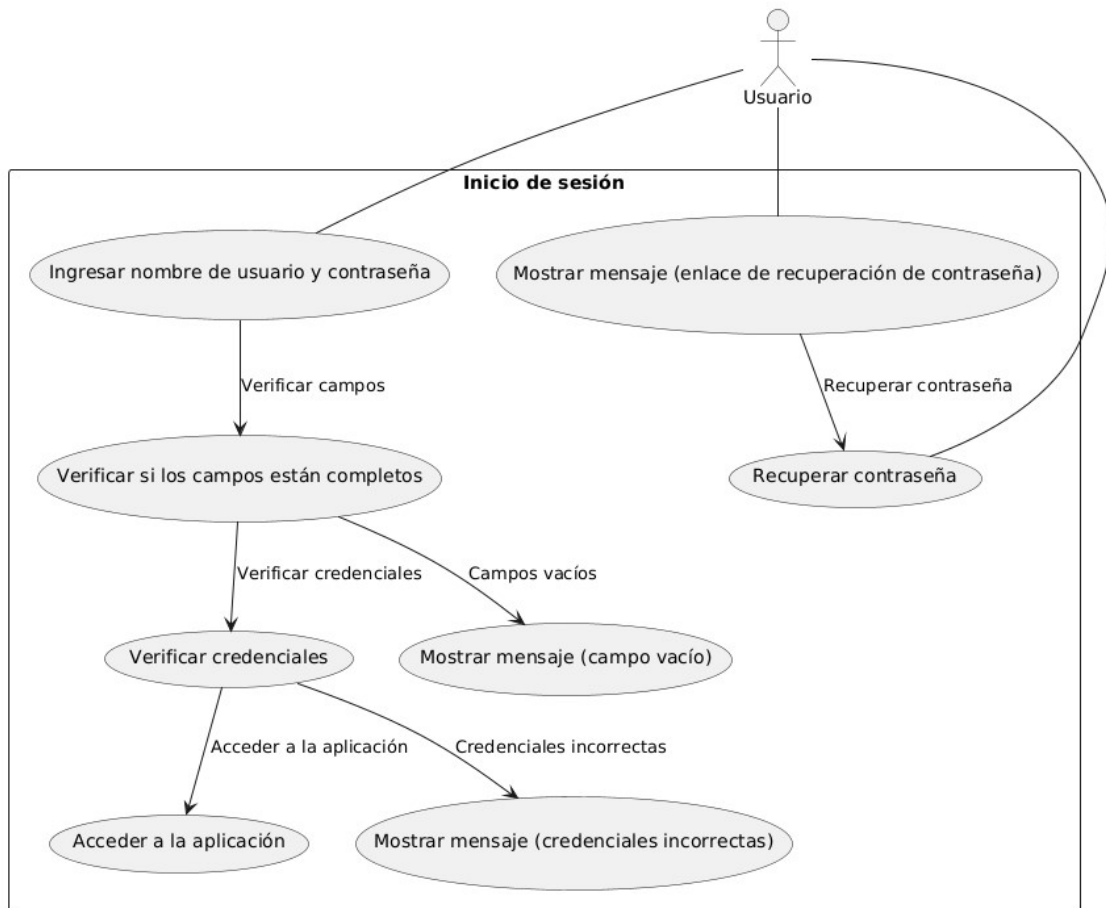
Actualmente existen algunas aplicaciones que ayudan a gestionar colecciones de videojuegos, como GG App, HowLongToBeat o Backloggd. Estas apps permiten realizar un seguimiento, pero no ofrecen funcionalidades como el almacenamiento físico. Además, todas son aplicaciones web, esta es una alternativa a nivel de aplicación para la gente que prefiera tener la aplicación a mano, y que no sea como puede ocurrir con las otras, que un fallo en la web puede evitar que accedes a tu biblioteca durante mucho tiempo. Aquí únicamente necesitas conexión a internet y que la web donde se aloja la base de datos funcione, no dependes de las web del servidor de la empresa o posibles ataques a la misma.

Mejoras de la aplicación respecto a las existentes

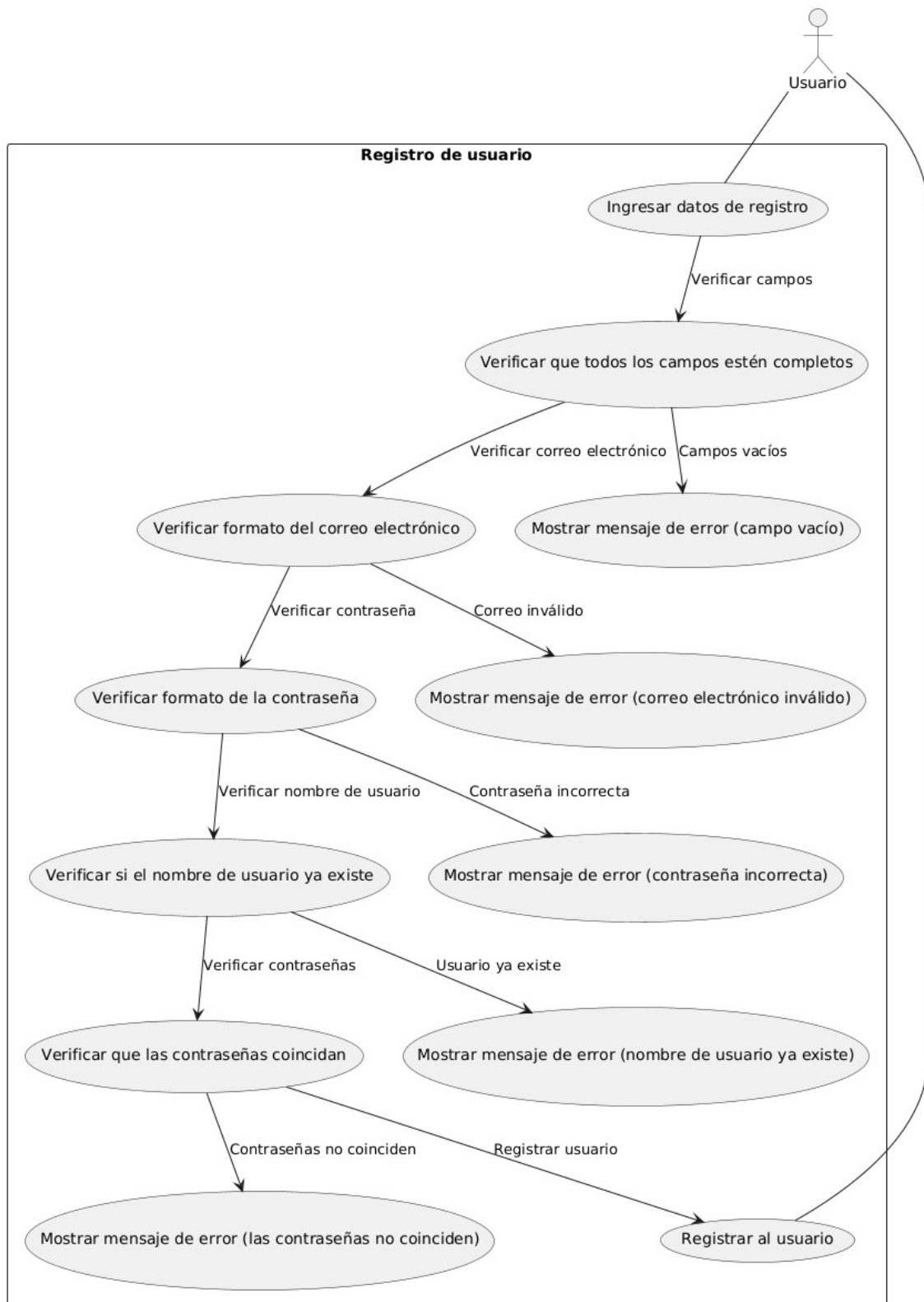
La aplicación permite localizar juegos físicos, mostrar estadísticas personalizadas (sobre nosotros, como los géneros más jugados, consola para la que más juegos tienes...), además de funcionalidades como exportar colecciones en PDF y conectar con bases de datos en la nube. Y como mencionamos anteriormente, no dependes tanto de factores externos como el servidor de una web.

Casos de Uso - F

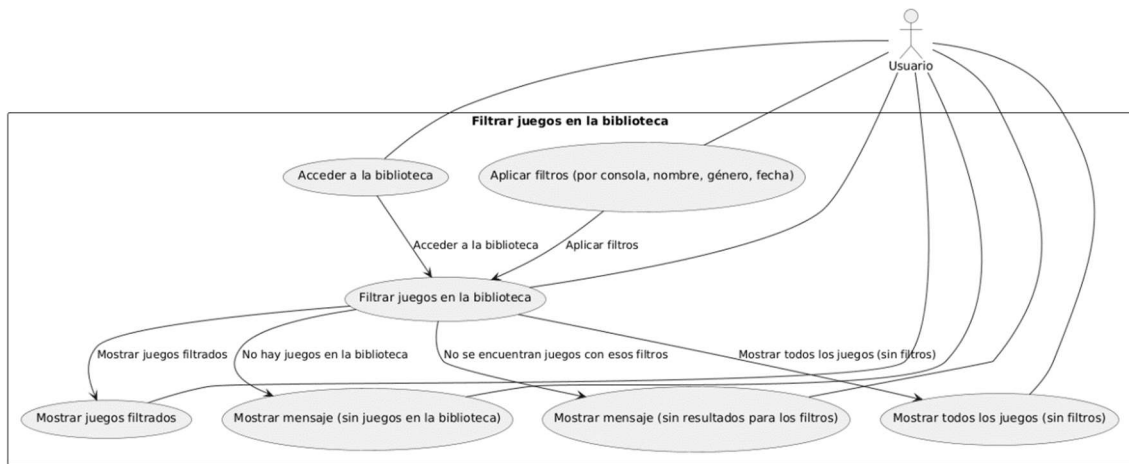
- Inicio de sesión



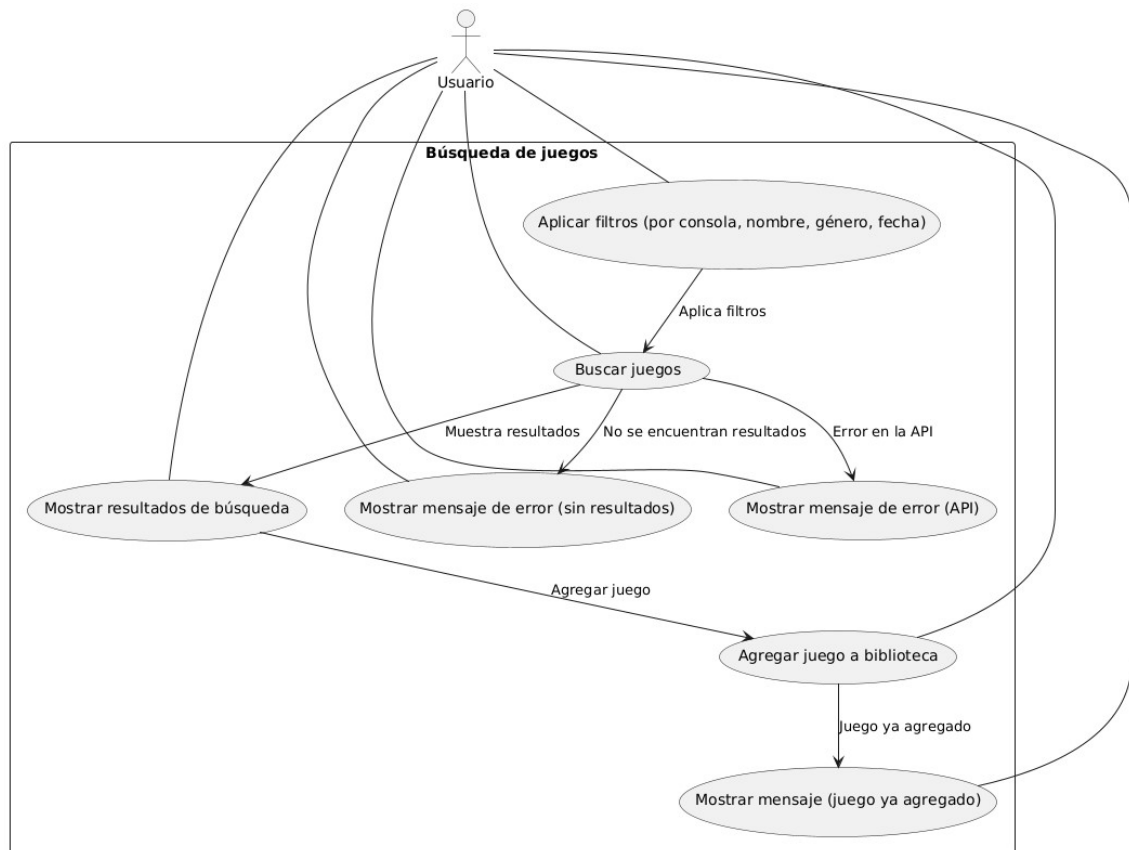
- Registro



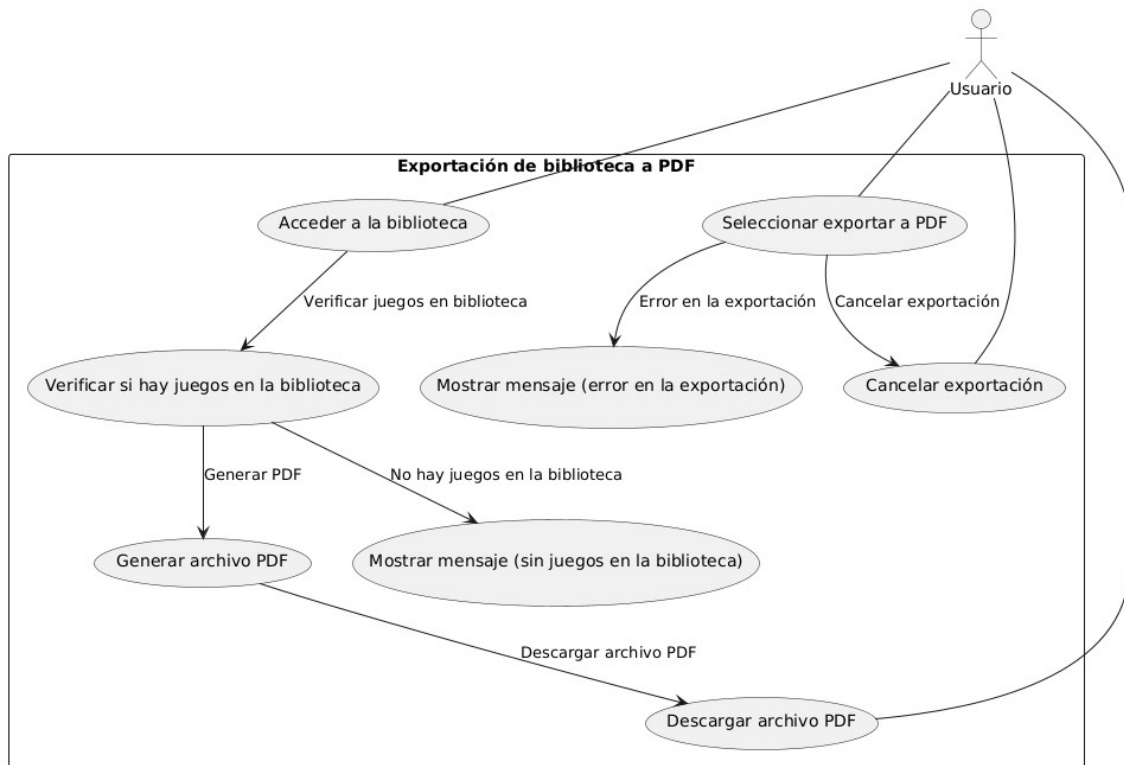
- Filtrar juegos en biblioteca (y wishlist)



- Buscar juego para biblioteca (y wishlist)



- Exportar biblioteca a PDF



Requisitos funcionales y no funcionales

Funcionales

- Login obligatorio
- Permitir guardar, añadir y eliminar (eliminar en progreso) videojuegos.
- Gestión de wishlist
- Estadísticas
- Exportación a PDF
- Cuentas con bibliotecas propias.

No funcionales

- Conexión a APIs
- Conexión a la BBDD en línea
- Contraseñas hasheadas
- Escalabilidad

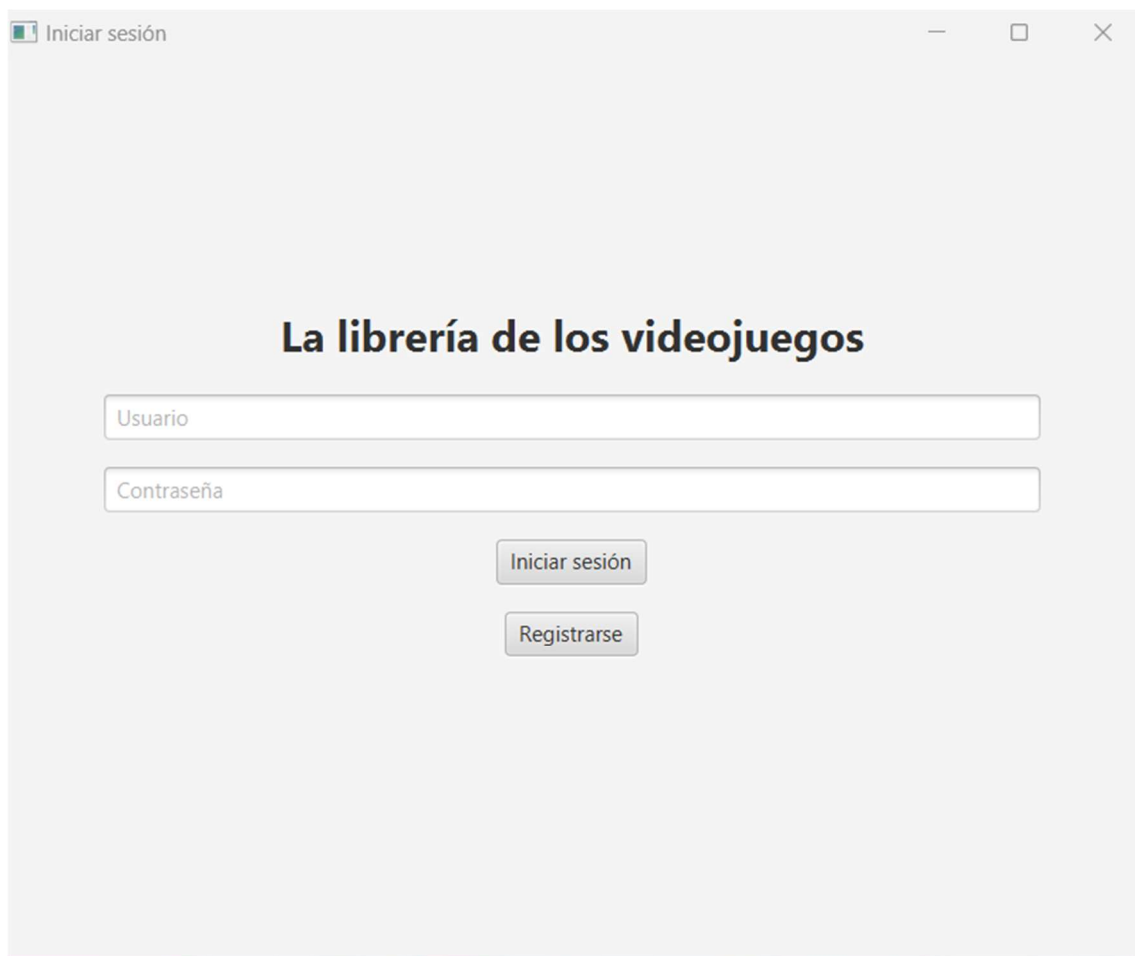
Diseño

GUI

UI (vistas)

Pantallas: login, registro, biblioteca, wishlist, estadísticas, detalle del juego. Las pantallas de Wishlist y estadísticas no se incluirán aún, pues deben ser modificados por la retirada de la funcionalidad de los precios. El resto de pantallas se mostrarán aquí (la de detalles se muestra dando un click al juego en la biblioteca del usuario, pero también va a ser mejorada ahora que todo funciona bien)

Pantalla login:



The screenshot shows a web application window titled 'Iniciar sesión'. The main heading is 'La librería de los videojuegos'. Below the heading are two input fields: 'Usuario' and 'Contraseña'. At the bottom of the form are two buttons: 'Iniciar sesión' and 'Registrarse'.

Iniciar sesión

La librería de los videojuegos

Pantalla registro:

Registro de Usuario

Registro de Usuario

Nombre de Usuario

Correo electrónico

Contraseña

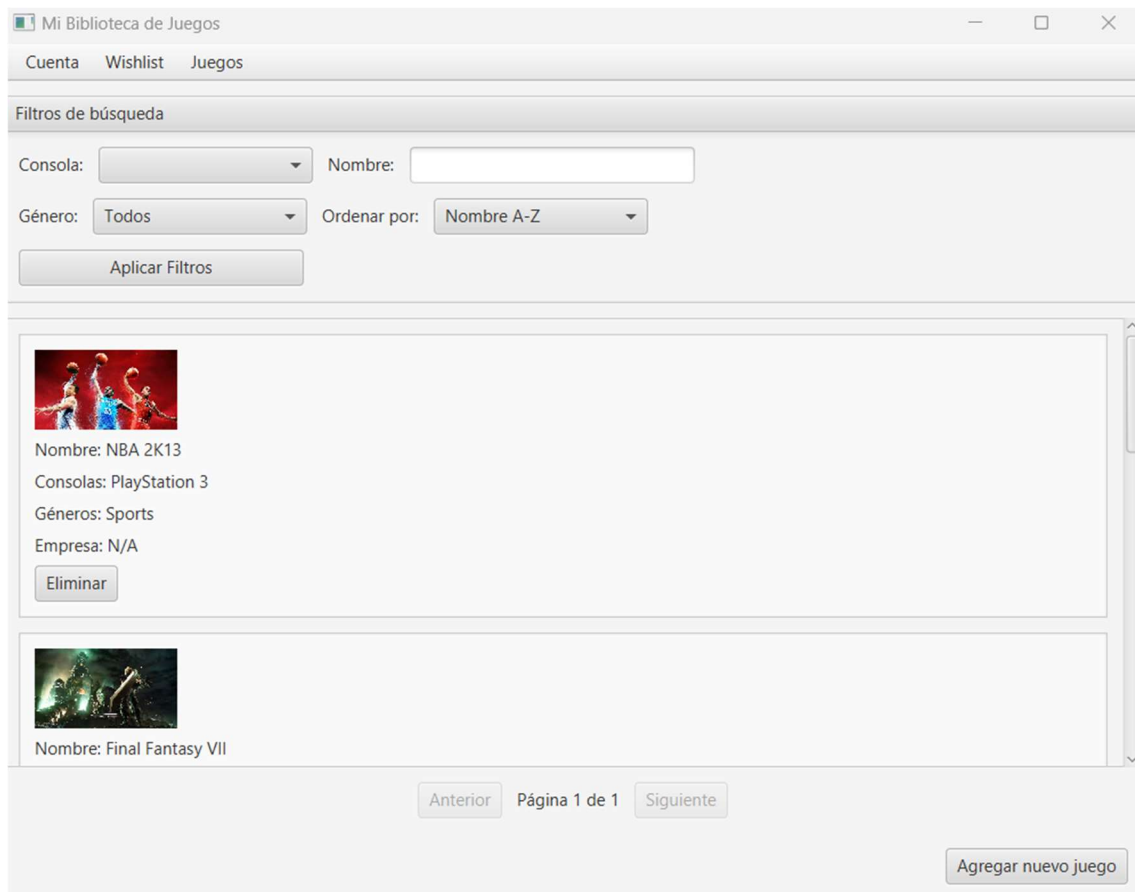
Confirmar Contraseña

Todos los campos son obligatorios.

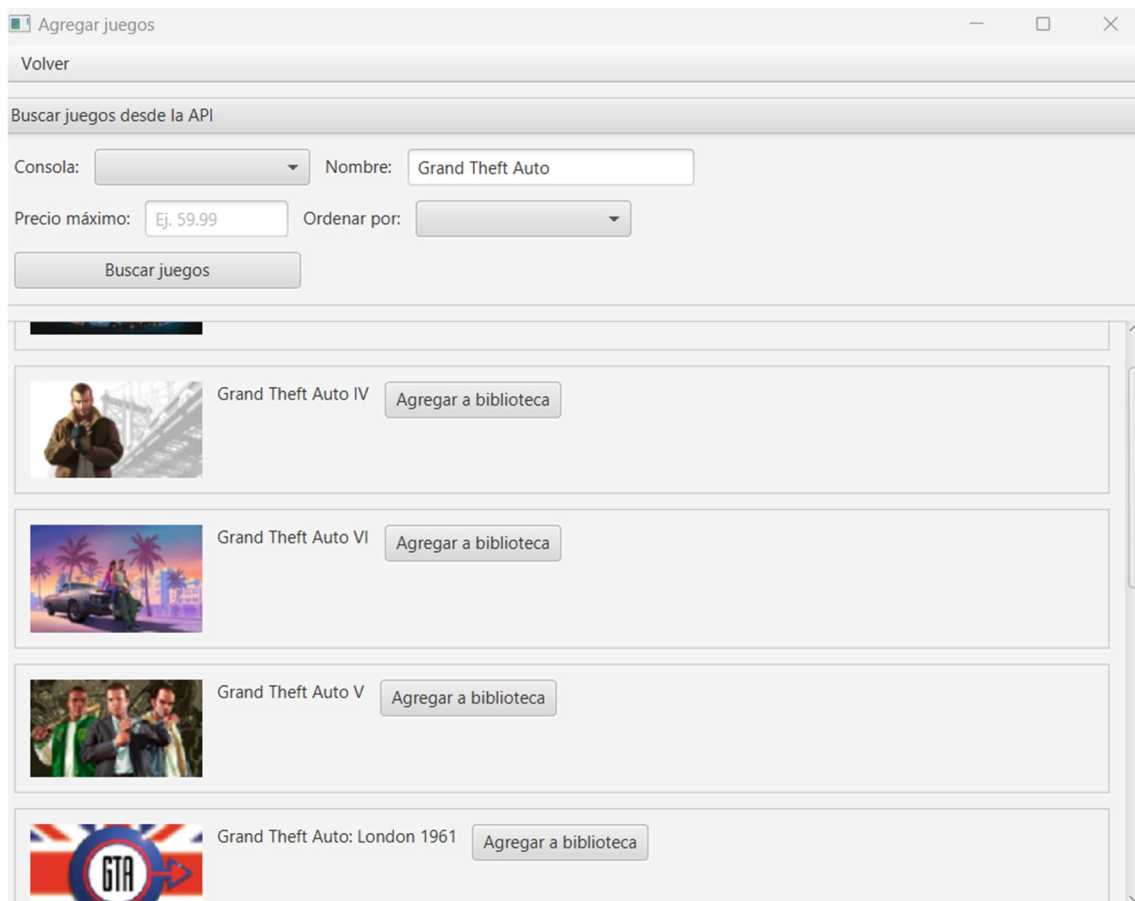
Registrarse

Volver a Iniciar Sesión

Pantalla biblioteca:



Pantalla búsqueda juegos (falta paginación, pero se añadirá, pues es copier y pegar la lógica de la biblioteca cambiando un par de cosas)

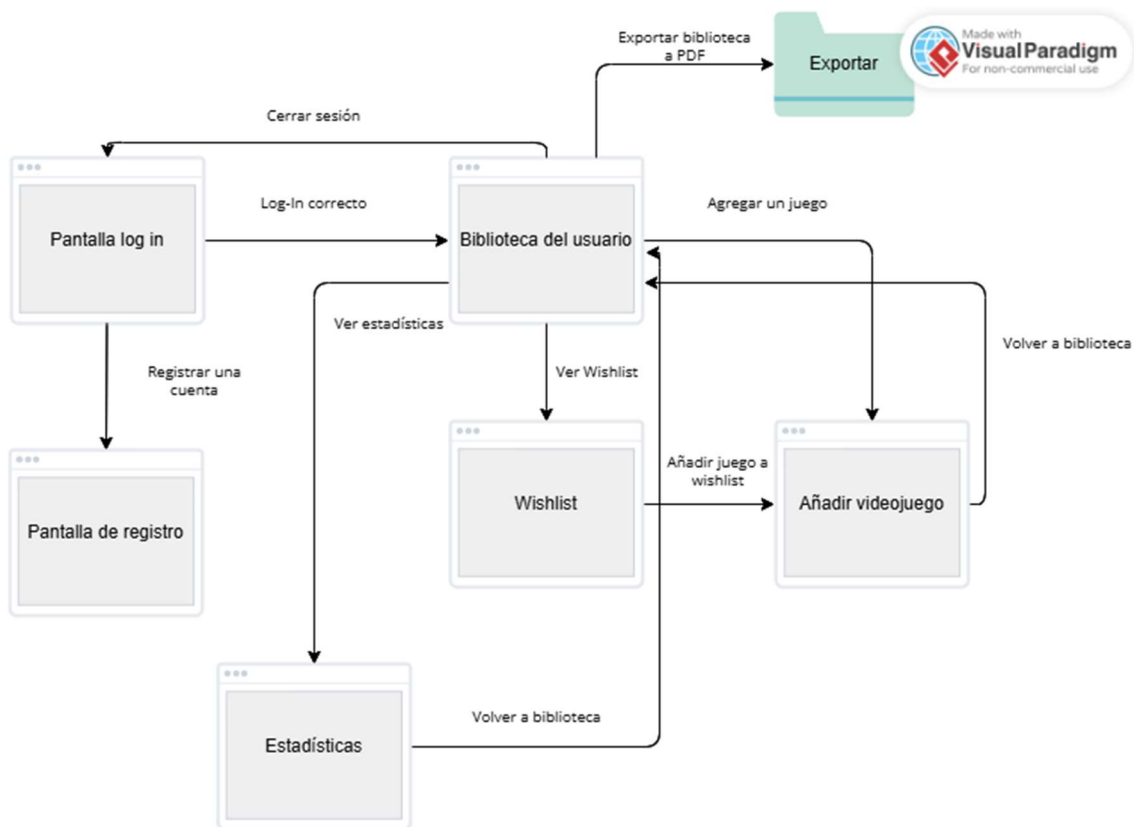


El resto de pantallas deben ser modificadas por complete debido a la retirada de la funcionalidad de precios por problemas con la api. La exportación de videojuegos no será una pantalla, si no una opción en la pantalla de biblioteca.

UX (usabilidad)

Diseño con JavaFX y Scene Builder, navegación intuitiva con controladores para cada pantalla.

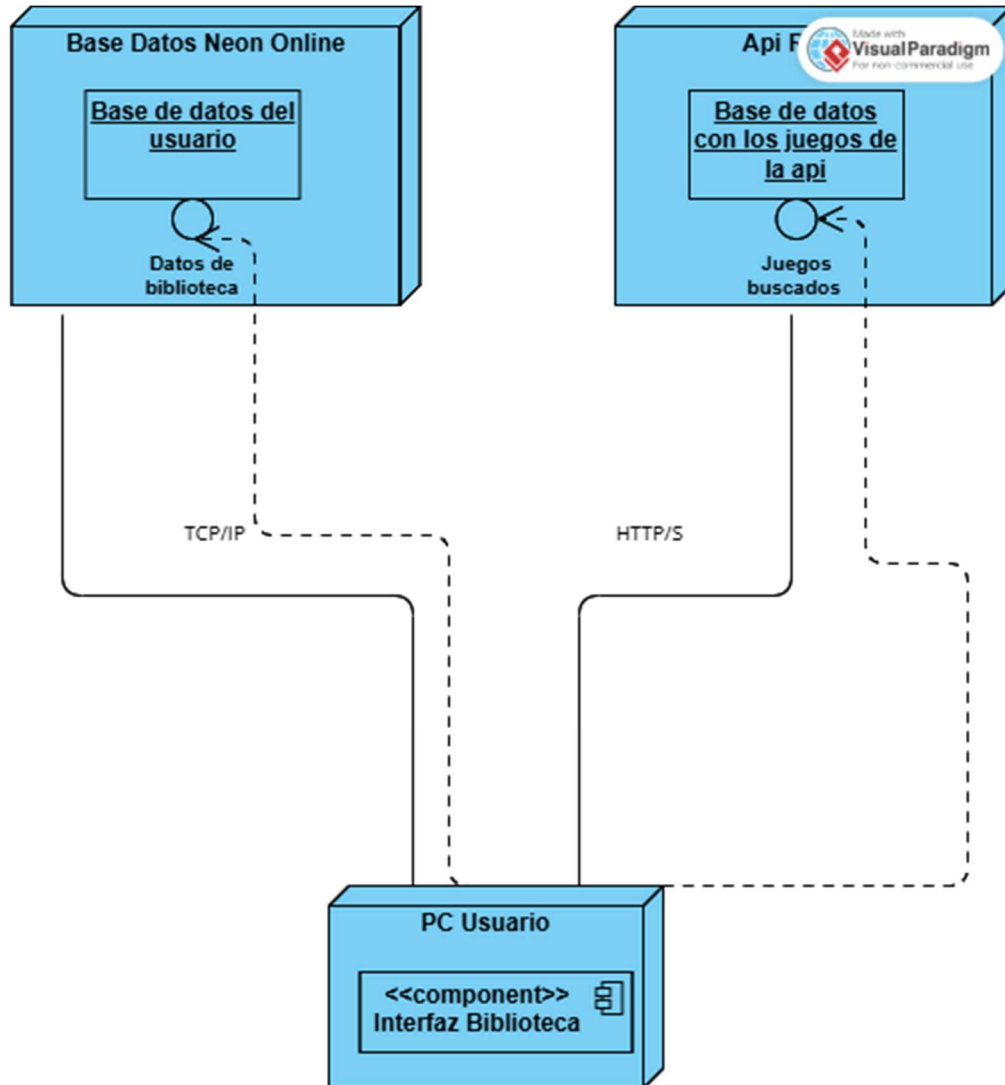
Diagrama navegación



Arquitectura

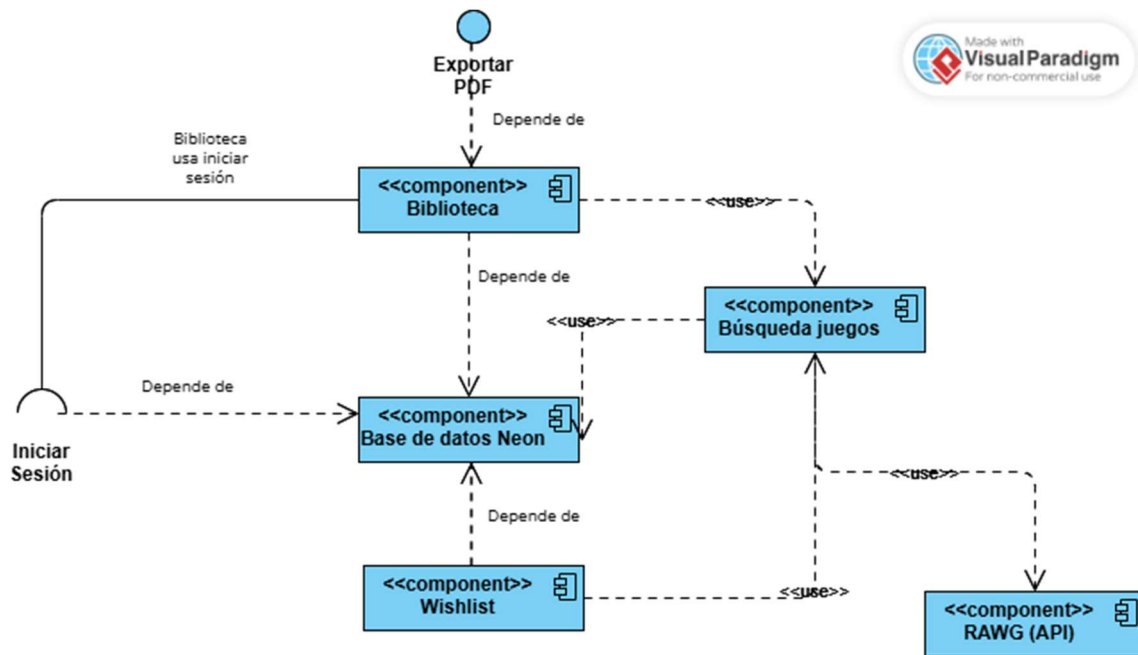
Despliegue

Aplicación de escritorio con base de datos en la nube y consultas a una api:



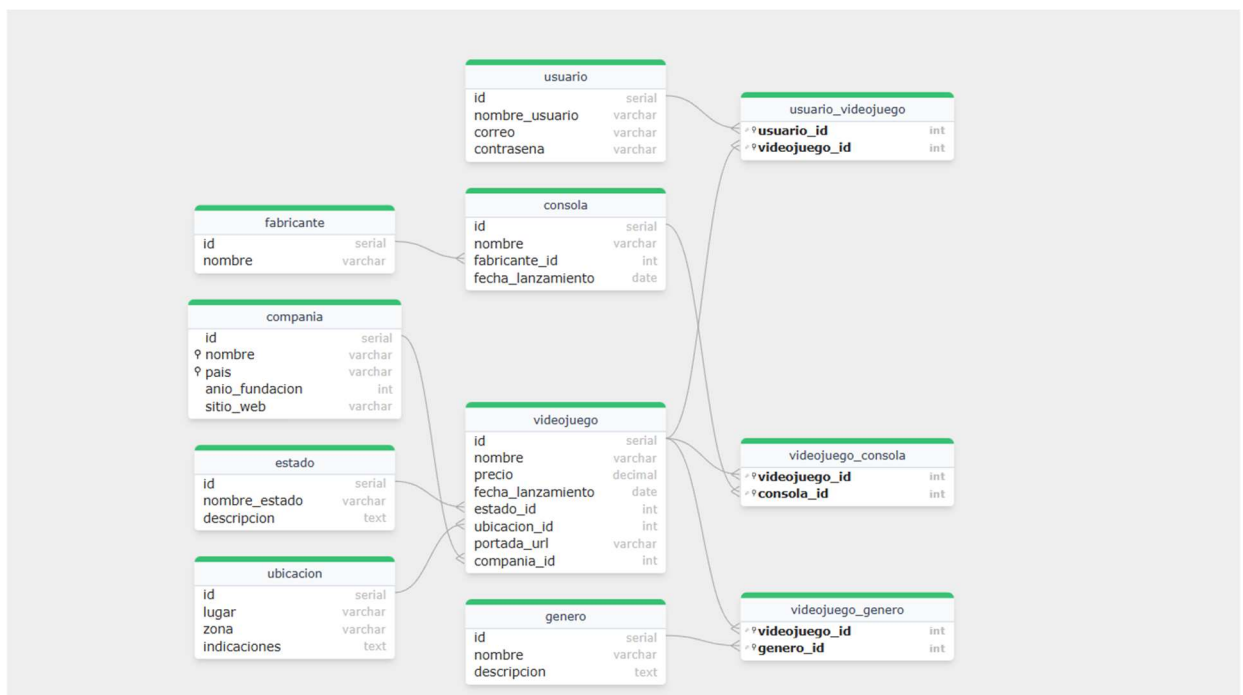
Componentes

MVC + Spring Boot. Backend arranca junto con JavaFX. También se ha desarrollado un cliente REST utilizando Retrofit para consumir la API de RAWG. Esta API proporciona información detallada y actualizada sobre videojuegos y es fundamental para llenar la base de datos del usuario sin necesidad de introducir manualmente todos los datos. Para el acceso a datos, se emplea Spring Data JPA con Hibernate como implementación. Esto permite abstraer las consultas SQL y aplicar principios de diseño como la separación de responsabilidades entre capas. La comunicación con APIs externas se gestiona mediante una arquitectura basada en Retrofit junto con OkHttp para el manejo de peticiones HTTP. Esto proporciona una forma robusta y mantenible de integrar servicios REST.



Base de datos

Base de datos PostgreSQL alojada en Neon. Se utiliza Hibernate como ORM de la aplicación, además de JPA para simplificar las consultas PostgreSQL (CRUD) además de anotaciones.



Paquetes, Interfaces y Clases

En este apartado, además de los paquetes clases e interfaces, me gustaría comentar el uso de Lombok para ahorrar código (constructores y getter y setters sobretodo)

Paquetes: api, dentro de api: retrofit, configuraciones, controlador, modelo, dentro de modelo: ids, repositorio, servicio, utils, resources, vista
Clases dentro de api: RAWGContenedorPlataforma, RAWGGenero, RAWGPlataforma, RAWGRespuesta, RAWGServicio y RAWGVideojuego.

Clases dentro de retrofit: RAWGCliente

Clases dentro de configuraciones: ConfiguracionBean y ConfiguracionRestTemplate

Clases dentro de controlador: BibliotecaControlador, ControladorBusqueda, LoginControlador y RegistroControlador (faltan clases por crear)

Clases dentro de modelo: Compania, Consola, Estado, Fabricante, Genero, Ubicacion, Usuario, UsuarioVideojuego, Usuario, Videojuego, VideojuegoConsola, VideojuegoGenero.

Clases dentro de ids: UsuarioVideojuegoID, VideojuegoConsolaID, VideojuegoGeneroID

Clases dentro de repositorio: CompaniaRepositorio, ConsolaRepositorio, EstadoRepositorio, FabricanteRepositorio, GeneroRepositorio, UbicacionRepositorio, UsuarioRepositorio, UsuarioVideojuegoRepositorio, VideojuegoRepositorio

Clases dentro de servicio: ConsolaServicio, GeneroServicio, RawgApiServicio, UsuarioServicio, UsuarioVideojuegoServicio, VideojuegoServicio

Clases dentro de utils: Sesion

Clases dentro de programa: App y JavaFXApp

Vistas creadas por ahora: pantalla_inicio, pantalla_biblioteca, pantalla_busqueda y pantalla_registro

Vistas pendientes de modificaciones: pantalla_wishlist y pantalla_estadisticas

Plan de pruebas (cómo)

Pruebas unitarias con JUnit, conexión con APIs, pruebas manuales.

SE AÑADIRÁ EL PLAN DE PRUEBAS CUANDO SE TENGA EN LA FASE FINAL.

Implementación

Entorno de Desarrollo

Java 22, pues es la versión que es más compatible con las dependencias que necesito utilizar. Visual Studio Code, ya que es un entorno conocido y que contiene todo lo necesario para probar

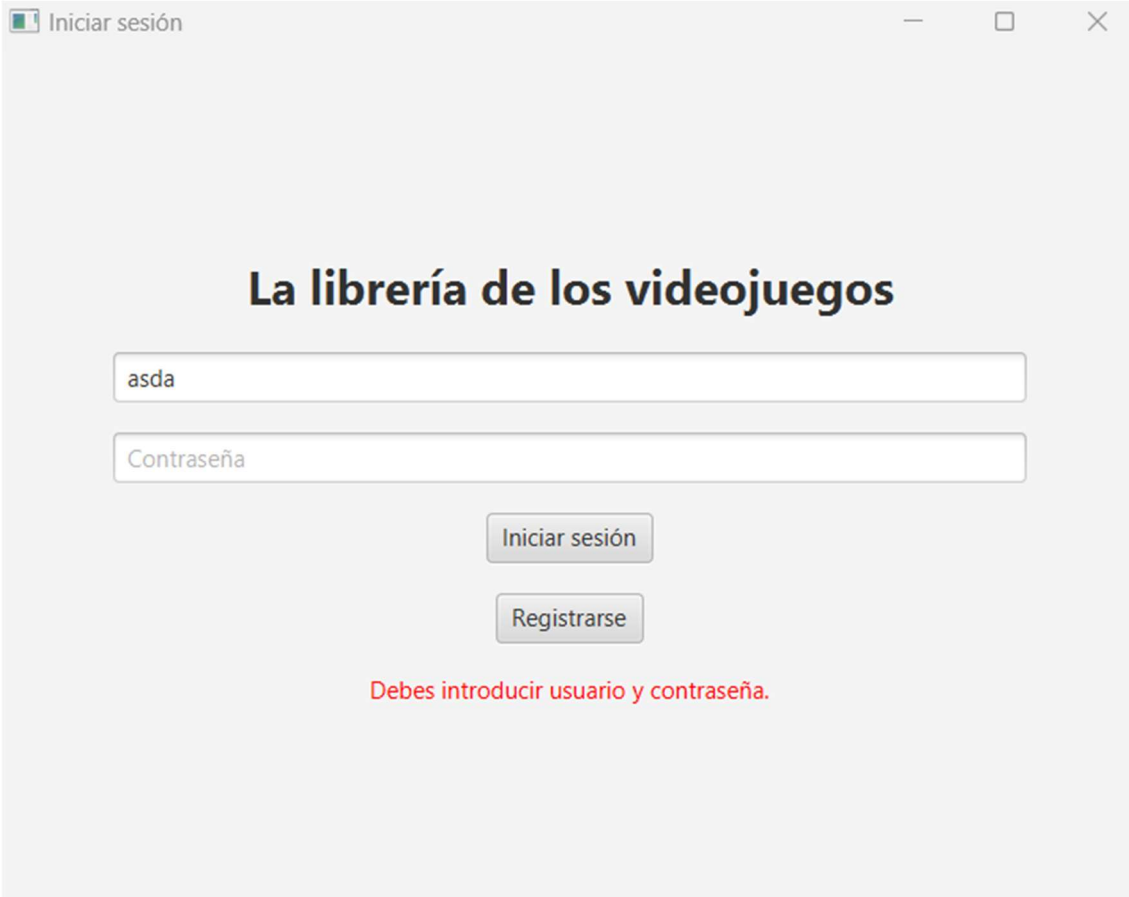
y desarrollar la aplicación, Maven para la gestión de dependencias y no tener que gestionárselas manualmente, Neon como sgbe pues usa postgresql, que es un lenguaje que funciona bien con JPA, aloja la BBDD on-line, para poder acceder a ella sin necesidad de tener la base de datos a nivel local, y tiene una versión gratuita que es suficiente para el Desarrollo de esta app y GitHub para llevar el control de versiones, además permite realizar backups para volver a una versión anterior. Para la exportación en formato PDF se usa OpenPDF, Jackson para el manejo de JSON (para las respuestas a las consultas a la api RAWG), Jakarta Validation e Hibernate Validation para las validaciones. Se usa Retrofit para el consume de la api.

Implantación/Puesta en producción

La aplicación se ejecuta localmente mediante Maven con conexión a una base de datos alojada en la nube. Cuando se desea añadir un videojuego, se realiza una consulta a una api con videojuegos (RAWG), se puede buscar mediante nombre, consola etc. Si pulsas el botón de agregar, se agregará a tu biblioteca. También tiene un sistema de log in que permite tener varios usuarios con sus propias bibliotecas.

Capturas de la ejecución de la funcionalidad

Introduciendo un usuario incorrecto:



La imagen muestra una ventana de navegador con el título "Iniciar sesión". El contenido principal es un formulario con el encabezado "La librería de los videojuegos". El formulario contiene dos campos de entrada: el primero tiene el texto "asda" y el segundo está etiquetado como "Contraseña". Debajo de los campos hay dos botones: "Iniciar sesión" y "Registrarse". En la parte inferior del formulario, hay un mensaje de error en rojo que dice "Debes introducir usuario y contraseña."

Iniciar sesión

La librería de los videojuegos

Iniciar sesión

Registrarse

Usuario o contraseña incorrectos.

Registrando un usuario con algún dato incorrecto:

Registro de Usuario

Registro de Usuario

Todos los campos son obligatorios.

Registrarse

Volver a Iniciar Sesión

Registro de Usuario

—

□

×

Registro de Usuario

El correo electrónico no tiene un formato válido.

Registrarse

Volver a Iniciar Sesión

Registro de Usuario

—

□

×

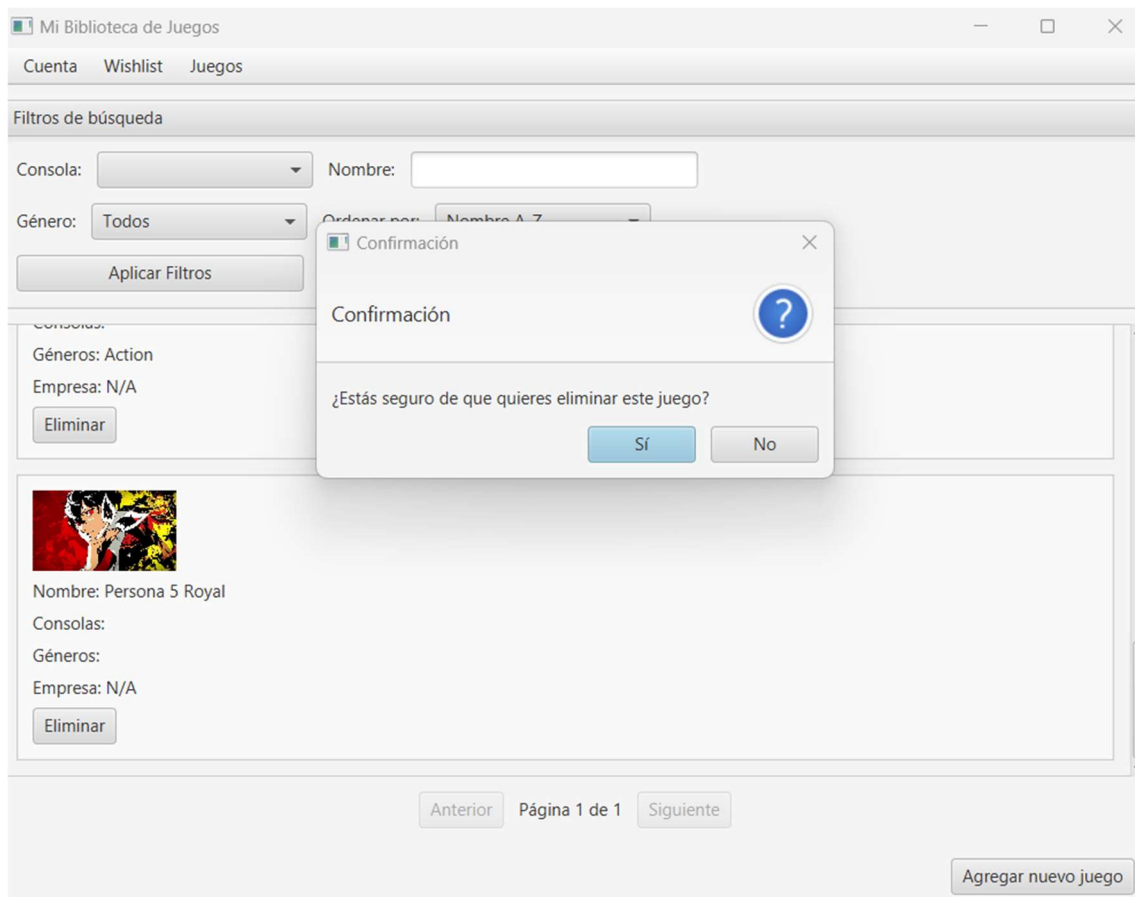
Registro de Usuario

La contraseña debe tener al menos 8 caracteres, una mayúscula, una minúscula, un número y un carácter especial.

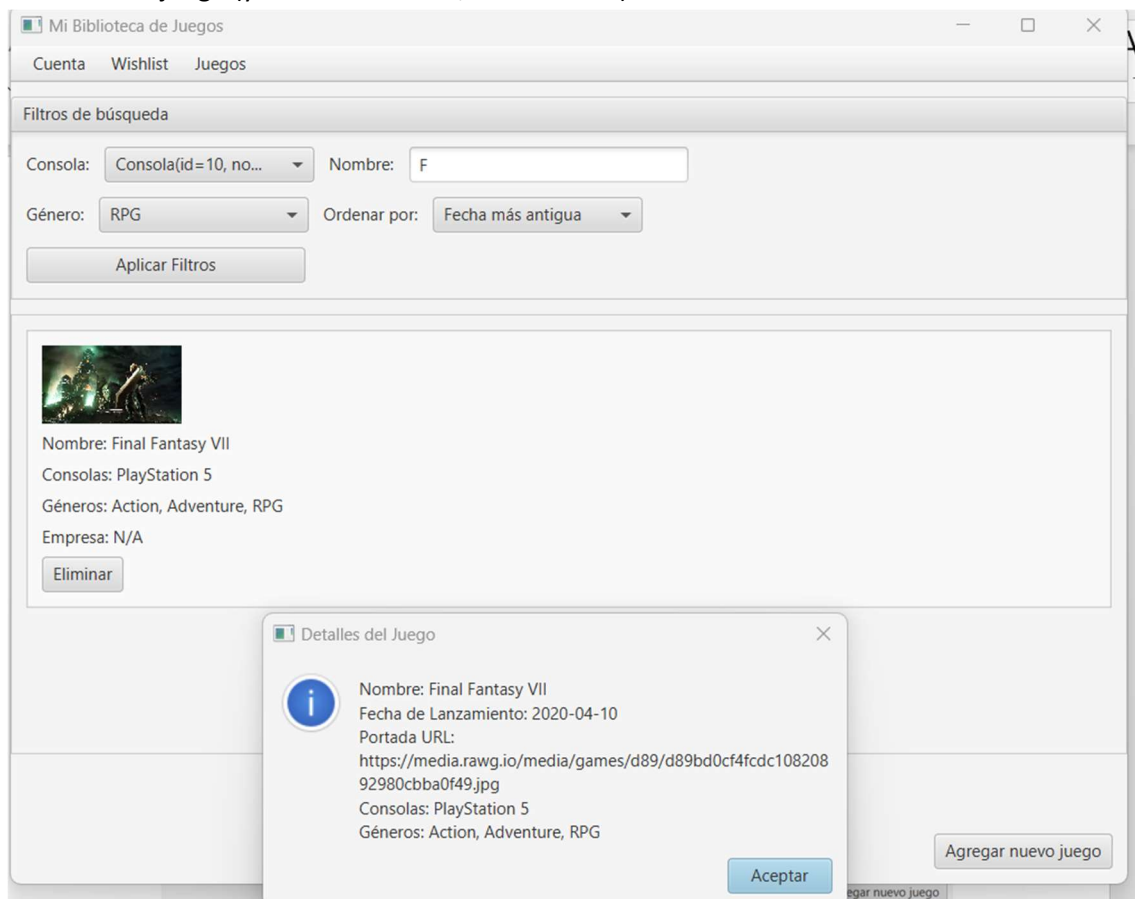
Registrarse

Volver a Iniciar Sesión

Eliminando un juego:



Filtrando un juego (y datos del mismo, en fase beta):



Buscando un juego para añadir:

Agregar juegos

Volver

Buscar juegos desde la API


Consola: PlayStation

Nombre: Pro evolution


Precio máximo: Ej. 59.99

Ordenar por: Fecha de lanzamie...


Buscar juegos

Pro Evolution Soccer 6


Agregar a biblioteca

Pro Evolution Soccer 2017

Agregar a biblioteca

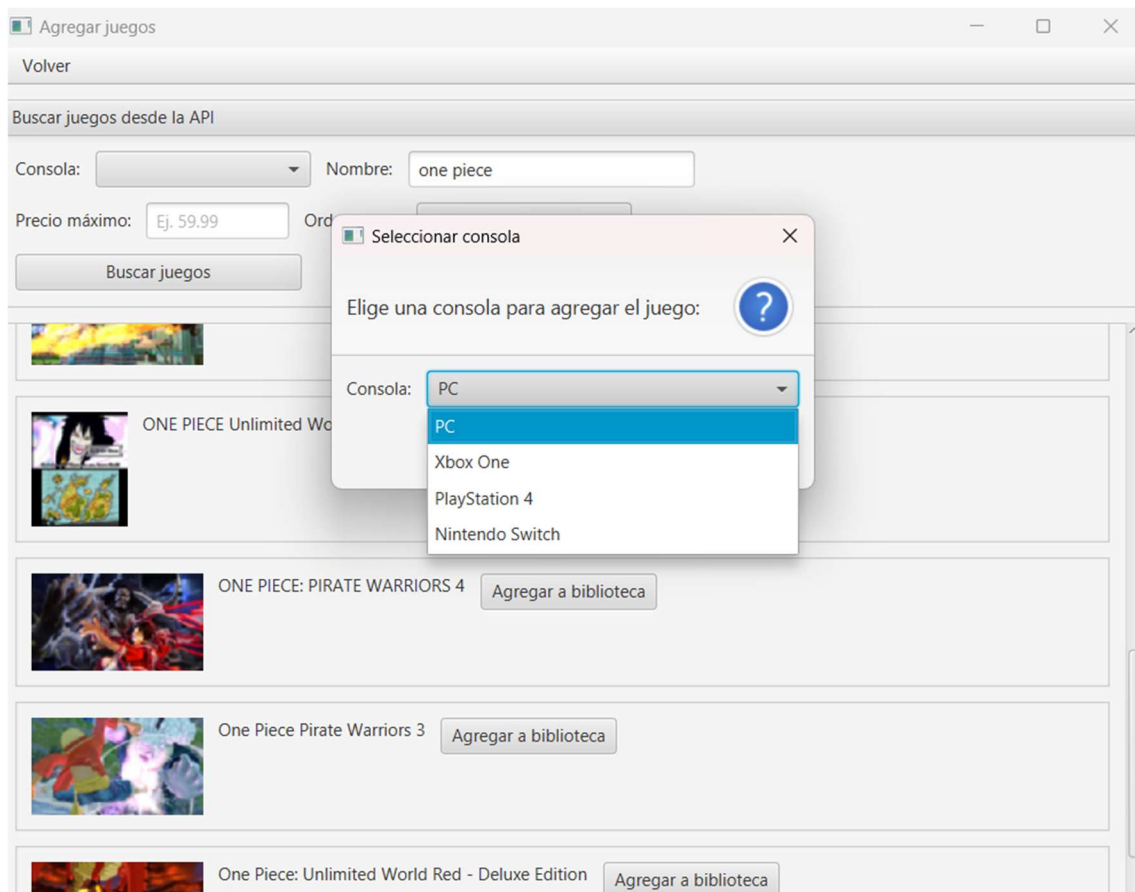
Pro Evolution Soccer 2009

Agregar a biblioteca

Pro Evolution Soccer 2014

Agregar a biblioteca

Añadiendo un juego:



Información sobre la versión y software necesario

- Java 22
- Visual Studio Code
- Neon DB
- Maven
- GitHub
- Scene Builder
- Conexión a Internet para uso completo

Elementos destacables del desarrollo

- Integración con APIs para la búsqueda de videojuegos y agregarlos en la bbdd, para no tener necesidad de crear una base de datos propia de donde añadir todos los videojuegos, si no ir rellenando nuestra base de datos extrayendo los datos de la api con los datos de los videojuegos.
- Base de datos en la nube para poder tener acceso a la misma en cualquier situación (siempre y cuando se tenga internet) y evitar problemáticas como perder la base de datos debido a un formateo o tener que reinstalar el SO.

- Estadísticas basadas en nuestra biblioteca y exportación de la biblioteca en formato pdf para poder mostrarla de manera sencilla.
- Uso de JavaFX moderno para poder tener unas interfaces simples pero funcionales
- Modularidad y escalabilidad en caso de querer añadir nuevas funcionalidades.

Manual de usuario

EN PROCESO CUANDO TENGA TODO MÁS O MENOS TERMINADO

Conclusiones

EN PROCESO

Bibliografía

- OpenAI. (2025). *ChatGPT*. Para consultas de errores y código básico Recuperado de <https://chatgpt.com/>
- Google. (2025). *Google Search*. Para buscar diferentes informaciones Recuperado de <https://www.google.com/>
- Gemini. (2025). *Gemini AI*. Para consultas de errores, usabilidad de interfaces y código básico Recuperado de <https://gemini.google.com/>
- Stack Overflow. (2025). *Stack Overflow*. Para preguntar y buscar respuestas en los foros Recuperado de <https://stackoverflow.com/>
- Neon. (n.d.). *Neon Docs*. Para consultas de cómo conectar la app a su BBDD y realizar operaciones Recuperado de <https://neon.tech/docs>
- Oracle. (n.d.). *Java Documentation*. Para dudas básicas de programación en java Recuperado de <https://docs.oracle.com/en/java/>
- RAWG. (n.d.). *RAWG Video Games Database API*. Para consultar lo que ofrece la aplicación en cuanto a datos y cómo usarla Recuperado de <https://rawg.io/apidocs>
- Visual Paradigm. (n.d.). *Visual Paradigm Diagram Maker*. Para la creación de diversos diagramas. Recuperado de <https://www.visual-paradigm.com/>
- PlantText. (n.d.). *PlantText*. Para la creación de los casos de uso utilizando código. Recuperado de <https://www.planttext.com/>

Anexos